Face Mask Detection using YOLOv5

Hosna Eltarras
Computer Engineering Department
University of Ottawa
Ottawa, Canada
helta037@uottawa.ca

Minah Ghanem
Computer Engineering Department
University of Ottawa
Ottawa, Canada
mghan079@uottawa.ca

Michael Khalil
Computer Engineering Department
University of Ottawa
Ottawa, Canada
mkhal120@uottawa.ca

Sarah Elmasry
Computer Engineering Department
University of Ottawa
Ottawa, Canada
selma083@uotttawa.ca

Shahenda Youssef
Computer Engineering Department
University of Ottawa
Ottawa, Canada
syous071@uottawa.ca

***Abstract*—After Covid19 had invaded our lives changing its whole shape, there was a crucial need for stopping the spread of the virus by automatic means. A solution that ensures people are abiding by the simplest safety measures as "wearing masks". In this project, we propose a pipeline where we train YOLO-v5 model with different hyperparameters on our collected dataset while applying augmentation techniques. Our best model reached high performance on only 700 images with precision, recall and mean average precision (mAP) of above 80%. The masks are detected whether they are upfront, sided or at moderate distances from the cameras. However, further improvements can be done on the model to evade the few cases it fails at.**

***Index Terms*—Covid-19, object detection, deep learning, face-mask, YOLO, augmentation, mAP.**

I. Introduction

Two years ago, the life of the whole world has been reshaped by the hands of COVID19. This reshaping was not manifested only in hospitals and medical sectors. However, it affected the different aspects of our lives. At first, there was this long period of lockdown to save people from being infected by the disease, nevertheless while saving people from death by coronavirus, other died out of poverty due to the harsh and rapid economic slump as a result of the lockdown. Thus, it was globally decided to evade the hazards of covid19 by means other than the lockdown like applying the safety measures everywhere. These measures should be imposed on everyone to impede the transmission of the virus as much as possible and they are mainly about social distancing and wearing masks, where here we are concerned only with "wearing masks" [1], [2]

While being easy to be complied with, many people who didn't lose their dear relatives or friends believed that Covid19 is just a mere hype and they can surpass it easily that it doesn't need abiding by such measures. This has appeared greatly at the first days of removing the lockdown especially among the youth, where they would deliberately shake hands and remove masks as if nothing has changed. That's why the authorities dedicated labours and guards to manage people's attitudes in order to hinder them from acting freely and making their lives along with others in danger, especially with the fact that some might transmit the virus without being affected with it or without showing any symptoms. Even though this has been managed to some extent at the beginning, we can't always rely on human beings to dedicate their eyes picking those who are not abiding by the rules in wearing masks. Moreover, given that currently a lot of people are unaware of the new Covid19 variants and think we have passed the catastrophe, we are in dire need of more automatic systems to manage this situation. Fortunately, such problem can be translated into one of the most classic problems in computer vision: Object Detection.

In this project, we tackle this problem of automatically detecting face masks by proposing a complete pipeline using the state-of-the-art algorithm in object detection which is YOLO-v5, in contrast to the classical model used in such problem: Mask-RCNN [3]. As the feasibility and durable computation of YOLO v5 that doesn't require costly devices perfectly matches our aim in developing a pipeline that can be easily incorporated in surveillance cameras.

The next sections of the paper shall go into more details reviewing the endeavours made in this area then discussing our pipeline starting from the dataset, the annotation and augmentation, the model used along with the different hyper-parameters and combinations experimented with the results of each. Lastly, we shall talk in the discussion about the further steps we are thinking of as improvement for this pipeline.

II. Related Work

In previous work, we can see different approaches that are presented for detecting correctly-worn face masks. In [4], we can see two different approaches to reach the aforementioned task. One approach is to detect masks directly, while the other is to detect faces first, then to classify if the face has a correctly-worn mask or not. Before discussing any of these approaches or any other approach in the literature, we believe that it is beneficial to give a brief background on face detection

approaches. The structure of this section will be as follows: we will firstly mention very briefly two different ways of how face detection was tackled in the literature (with and without deep learning). Then we will go over an architecture of one of the deep learning approaches, specifically the general architecture of YOLO (You Only Look Once); after that we will go over some face mask detection approaches in the literature.

A. Face detection

For face detection, we can see some approaches that did not involve deep learning techniques such as the HAAR cascade algorithm mentioned in [6]. In HAAR, a set of filters are applied in stages to extract features, mainly edges and lines. The intuition is that some of these features will represent facial structures. To distinguish these features that represent facial structures from other features and to classify faces, Adaboost algorithm is used.

With the emergence of deep learning, more recent approaches use deep learning for face detection. That can be seen in the usage of architectures such as the R-CNN family and YOLO, as seen in [5], and [4] respectively. We will go over the architecture of YOLO in a very brief manner since that would be beneficial to the flow of our work. The architecture mentioned in this discussion is based upon our understanding of the original YOLO architecture mentioned in [7]. For more details about YOLO and R-CNN refer to [7] and [8] respectively.

B. YOLO and Object Detection

In YOLO, the problem is modeled as a regression problem. The image is divided into an SxS grid. A set of B boxes, called anchor boxes, is defined, where the intuition is that each anchor box would represent an aspect ratio of the aspect ratios of the set of objects that we are detecting. For example if we are detecting persons and cars, we could have two anchor boxes, one larger in height for the person object, and another one larger in width for the cars object since the first anchor box would not match well with the aspect ratio of a car. YOLO uses these anchor boxes to search for where these anchor boxes may be located in each grid cell, where each bounding box would result in 5 outputs for each cell: x and y coordinates of the location of the anchor box relative to the cell, width w and height h of the located bounding box, a confidence score of how confident YOLO of the existence of that box, where it should map to the value of the Intersection Over Union (IOU). For each cell, YOLO also predicts the probability of existence for each class object. That result of having the output of the YOLO network is to be $SxSx((5*B)+C)$ where C is the number of objects that we are detecting.

As understood from [10], in state-of-the-art object detection algorithms, the architecture mostly consists of three elements: Backbone, Neck, and a Head. The backbone is responsible for extracting basic features, the neck is responsible for extracting more elaborate features, and the head is where the network uses these features to reach an output. The head of the network could be a one-stage detector or two-stage detector. We would not go into much details of how they differ since that is out of the scope of this paper, but generally, one stage detectors are faster while two stage detectors are more precise. The reason for mentioning this part is that YOLO is a one-stage detector network while R-CNN is a two-stage detector network, which makes it easier for YOLO to be generally faster than R-CNN while R-CNN may be generally more precise.

C. Mask Detection

In [4], the first approach uses YOLO-v4 to detect two objects: one is a face without a mask and the other is a face with a mask. The second approach is to use YOLO-v4 to detect faces in general, with or without masks, and feed these detected faces to a classifier to determine if the face is a masked face or a non-masked face. For the classifier part, two custom CNN networks are used and their architectures are mentioned in the paper. One is a simple CNN network and the other is the same network but with some modifications to make it more complex. The dataset used is a combination of benchmark dataset and a novel dataset that were collected by the authors that fit well with the Iranian culture so that it could work well with faces wearing hijab, for example. Image augmentation of random perspective transformations, gaussian noise, and change in brightness were used. The results show that the second approach - YOLO-v4 followed by a classification network - combined with the complex network works on detecting faces with masks vs. faces without masks.

In [9], they used two architectures: YOLO-v3 and Faster R-CNN. The detection task included two objects: mask vs. no mask. They took advantage of transfer learning since the data they used was small. The data consisted of a combination of a benchmark dataset for mask detection along with a manually collected and annotated dataset. To train the models, the authors freeze all the layers except for the final layers and train the model till convergence, then they unfreeze all the layers and re-train until convergence. The conclusion reached is that Faster R-CNN has higher average precision while YOLO-v3 has a faster inference time and supports higher frame rates. The authors claimed that when observing the speed/accuracy trade-off, YOLO-v3 was better and hence can be used for real-time systems, but if high-end GPUs are available, one may go for Faster R-CNN.

In [3], the authors used Faster R-CNN, YOLO-v3, and YOLO-v3 with a change in the loss function used. This change in loss function is due to the authors' finding that the network is converging better in bounding boxes but not as much for the objects' classification. Therefore, a weighted loss function was used. This time there were three objects to detect: faces without masks, faces with masks and faces with incorrectly worn masks. Data augmentation, undersampling, and hyper-parameter tuning were used. The results findings

was that, in terms of Precision: Faster R-CNN performed best (0.932), followed by the weight loss YOLO (0.919), followed by YOLO (0.905). In F1-score: YOLO with weighted loss was better (0.731), followed by Faster R-CNN (0.721), followed by YOLO (0.706). And in terms of speed: YOLO with weighted loss was the fastest (95.78 ms), followed by YOLO (96.42 ms), followed by Faster R-CNN (175.2 ms). The time is measured per 100 images, presumably at inference time since the authors did not specify.

In this work, we use a YOLO version from a GitHub repository by ultralytics [13], that is, whether it is referred to as YOLO v5 or not is controversial. In this work we will refer to it as YOLO-v5. The detection task in this work is concerned only with one class to be detected, which is a face with correctly-worn face mask, while regarding all other cases as negative -whether it's an unworn mask or uncorrectly worn one-. The reason we chose YOLO-v5 is that as per our review, we didn't find any work in the literature that uses YOLO-v5 in face mask detection, and we are not aware of any published work that uses YOLO-v5 for this problem.

III. Dataset

A. Dataset structure

The dataset is collected by the authors in the wild, it is the production of our manual photo shots. Our data focused more on the youth images since they are the ones more expected to remove their masks or act casually. It was, also, captured in study places, restaurants and open air places commonly visited by the youth. The dataset accounted for the culture it was taken from, since the country had women with Hijab and without, so the dataset had images covering both categories as a similar work here [4]. Moreover, we took into consideration the ratio between male and female. We also accounted for the different coloured-masks to avoid the model from learning on only one colour of the mask.
In general, the dataset was ensured to be as clean as possible while having many variations in the brightness, contrast and angles to allow the learnability of the model under several conditions along with good learning from the clean data.

A total number of 500 images were collected then split by a ratio of 80%-20% among the training and validation/test sets, where 400 images were used for training, 95 images for validation, and 5 for testing.

B. Annotation

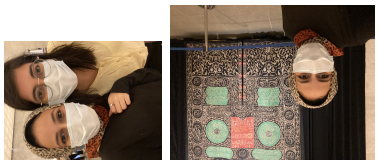
We have gone through different data annotation platforms until we settled on KILI technology [11] to create the bounding box labels. The images are labeled with 4 points to draw a rectangle around each face that has a mask worn correctly. Lastly, the box's 4 points were converted to YOLO labeling format which comprises the annotation of the corresponding image file, that is (1) object class, (2) object coordinates depicted as x-center and y-center, (3) height and width.

(a) (b)

Fig. 1. Images with people wearing the mask incorrectly.

(a) (b)

Fig. 2. Different angles for the images and sided face-masks.



(a) (b)

Fig. 3. Different lighting and various places.

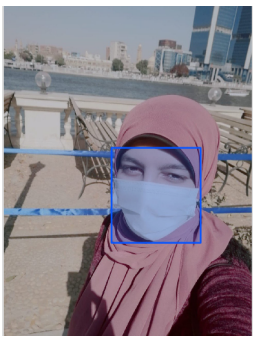


Fig. 4. Annotation Example

C. Data Augmentation

Since we have a very small dataset, we applied different data augmentation techniques. Data augmentation is the process of increasing the amount and diversity of data by transforming the existing training images to multiple forms.
The images are augmented using Gaussian noise, brightness manipulation, and blur which increased the size of the training dataset to 700 images.
Gaussian noise will help training the model on noisy data to generalize well on unclear noisy datasets. The Blur technique can be useful in making the model more robust to image quality issues. Brightness manipulation by Increasing and decreasing the brightness level of the image to help the model to detect the person with mask in different surroundings.

Sample of the applied augmentation techniques shown in Fig.[5].

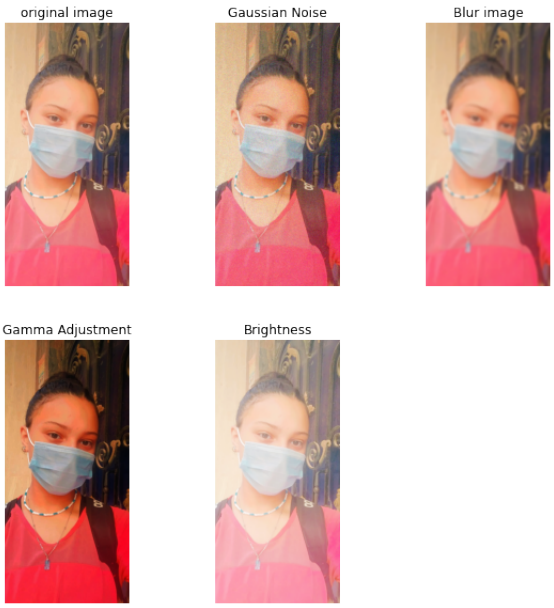


Fig. 5. Data Augmentation.

IV. Proposed Method

A. Pipeline

Our proposed solution's steps are mainly:

- Data collection
- Data annotation method
  We discussed the first two steps in section III.
- Apply YOLO Network
  We used YOLO V5 Network with selected label 'with mask' as we mainly focused on detecting the mask.
  Split the dataset into training, validation, and testing datsets.
  Tuning the model to detect the best combination of parameters that achieve high performance.
  Train and evaluate the model with the selected parameters.
- Data Augmentation: Train the model with the augmented data and evaluate the performance of the model.

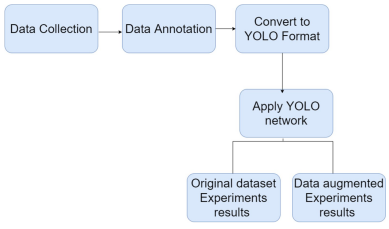


Fig. 6. General diagram for our proposed solution

The next section shows the results of the trained models and the comparison between the performance of the model with and without data augmentation.

B. Experiments

For our testing environment, we used Google Colaboratory (Colab) [12] to train and test our model as it gives us free access to GPUs, free storage and the advantage of easily shared notebooks.
Our collected dataset takes a space of 0.535156 GB to be stored. Our best model, which takes 12 epochs to coverage, takes 0.615 hours to be trained. Also, it takes 0.6ms to pre-process, 109.3ms inference, 1.5ms NMS per image for the inference time.

For evaluating our model in the experiments, we depended on a set of parameters which are :

1) Precision: the probability of predicted bounding box to be true.
2) Recall: the probability of the ground truth to be correctly predicted.
3) Mean Average Precision (mAP): the mAP parameter averages the average precision (AP) over all the classes. The average precision is used as a parameter to summarize the precision-recall curve. It summarizes both the precision and recall in one parameter.
4) The number of correctly predicted test samples: We have got a test sample of size 5, so having a score of 5 means that our model managed to detect all the test samples correctly.

To tune our model in the experiments, we have worked over a set of parameter values for each attribute:

1) For the image size, we have tried image sizes: 640 and 320.
2) For the number of batches: 16 and 8 batches.
3) For the number of epochs: we have tried 9, 12,15 epochs.
4) For the "Confidence threshold" we have set it to a constant value (0.25). The Confidence threshold is considered as the minimum confidence score the model has on a detected object.
5) "Iou threshold" is also set to a constant value(0.45). Iou is the intersection area over the union area of the overlapping bounding boxes. It determines a threshold for the area of intersected objects.

Table 1 represents our initial experiments, The results are most of the time similar. However, we can conclude that increasing the model's epochs helped it predict more correctly. Using the number of epochs as 9 did not help the model. Also, most of the time, using smaller numbers for the batch size such as 8 helped the model to score better results than using a batch size of 16. The image size parameter would both depend on and influence the use of other parameters. But generally, high image sizes such as 640 would influence better results.

From the table 1, we can say that our best model is the one having parameters of 640 image size, batch size of 8 and 12 epochs.

Table 2 represents the results of the best model from our initial experiments versus the best model with data augmentation added. The precision is increased by 1%. The recall and

Image Size	320						640					
Epochs	9		12		15		9		12		15	
Batch Size	8	16	8	16	8	16	8	16	8	16	8	16
Recall	0.8	0.73	0.98	0.86	0.95	0.97	0.81	0.79	0.97	0.89	0.97	0.97
Percision	0.66	0.64	0.71	0.64	0.79	0.69	0.75	0.62	0.8	0.75	0.81	0.78
mAP	0.71	0.649	0.8	0.7	0.83	0.78	0.77	0.7	0.84	0.79	0.81	0.83
Correctly Predicted Test Samples	3	3	4	3	4	3	3	3	4	4	4	4

TABLE I
TUNING RESULTS.

	Without Data Augmentation	With Data Augmentation
Parameters	Image Size = 640 Batch Size = 8 Epochs = 12	
Percision	0.973	0.983
Recall	0.801	0.838
mAP	0.842	0.874
Correctly Predicted Test Samples	4	4

TABLE II
COMPARE MODEL WITH AND WITHOUT DATA AUGMENTATION

the mAP are both increased by 3%. This means that the data augmentation is promising as it helped our model to perform better and generalize on more examples.

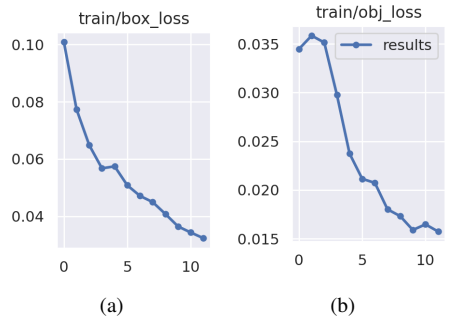


Fig. 7. Training Loss for the detected object and the bounding box.

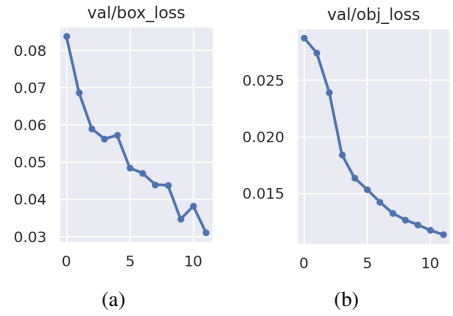


Fig. 8. Validation Loss for the detected object and the bounding box.

The figures 7 and 8 represents the validation and training losses across all the epochs. The loss is calculated for both: the detected object class and the predicted bounding box. As the number of epochs increase, the validation and training loss decrease. We have also tried to increase the number of epochs to more than 12, but the results started to be worse on the validation set.

Our best model succeeded in predicting four out of five images from our test samples. The sub figures of figure 9 represent the predicted bounding box for the test samples and the annotated bounding box. All our samples were correctly detected, except for (b) as part of the mask is not shown in the image.

Fig. 9. Predicted Bounding box vs Annotated bounding box in our test sample.

For further error analysis for our model, we have tried other test samples out of our dataset. These testing examples contain sided faces wearing masks, incorrectly worn masks, and people wearing masks from a distance far from the camera. From these results, we can see that our model succeeded in detecting sided faces wearing masks(a). People who wear the mask incorrectly like (c) were correctly ignored by the model, however the model did not ignore it in (b). For people at distance (d), the model somehow succeeded in detecting some of the far distanced faces, nevertheless its performance still needs to be increased. Finally, example (e) showing bare masks with no people wearing them, was correctly ignored by the model. As from our perspective, it is useless to detect unworn masks.

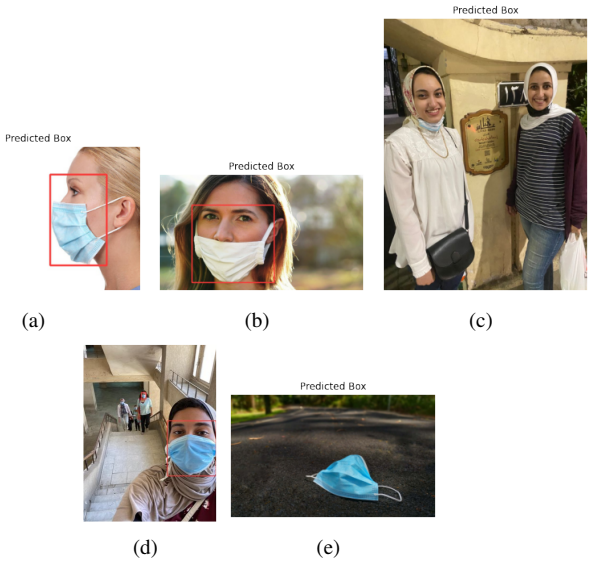


Fig. 10. Predicted Bounding boxes on other test samples.

V. Disscussion

As far as we have seen, the model is performing well on our validation set, however it isn't generalizing well on the new testing set. The model could detect that some masks are incorrectly worn. While in other cases where the mask was close to the noise but not covering it, the masks were wrongly detected by the model. Same applies to people far away from the camera. We could say that this problem arises because of our small dataset.

Future work may be to collect more data with various conditions and different orientations of people wearing masks. Also, using positional augmentation like cropping, scaling, horizontal and vertical shifting could help the model to detect and generalize on more scenarios. For example, the cropped face mask that the model failed to detect in the testing set could be detected after these modifications.

VI. Conclusion

In this paper, we have presented our work in detecting face masks using YOLO-V5. We have also presented our efforts in collecting diverse images to make our final dataset as far as we could. In which people were wearing masks correctly and incorrectly, multiple people in one image, and masks with no one wearing them.

A whole preprocessing pipeline was made to annotate and split our images for preparation in order to be fed to the model. Moreover, a lot of experiments were made to tune our model and find the best parameters. We have also increased our model's performance by using data augmentation through changing the brightness, adding noise, and blurring the images. Also, error analysis on the experiments wwas performed and investigated to define the model's weaknesses and define our next steps to be considered for future work.

VII. Acknowledgment

This section is to briefly mention and acknowledge the authors' contribution in the project. All the authors contributed in the dataset collection and literature review phases. The annotation and YOLO-formatting of the data was maintained by Sarah El-Masri and Michael Khalil. The data augmentation along with its relevant experiments were handled by Shahenda Youssef. Lastly, the integaration of the model with its different hyperparameter tuning was handled by both Minah Ghanem and Hosna Eltarras. Lastly, we do acknowledge the continual guidance from professor Jochen Lang and the TAs in our project.

References

[1] E. Nasiri, M. Milanova and A. Nasiri, "Video Surveillance Framework Based on Real-Time Face Mask Detection and Recognition," 2021 International Conference on INnovations in Intelligent SysTems and Applications (INISTA), Kocaeli, Turkey, 2021, pp. 1-7. doi: 10.1109/INISTA52262.2021.9548475

[2] K. N. S. Kumar, G. B. A. Kumar, P. P. Rajendra, R. Gatti, S. S. Kumar and N. Nataraja, "Face Mask Detection and Temperature Scanning for the Covid-19 Surveillance System," 2021 International Conference on Recent Trends on Electronics, Information, Communication Technology (RTEICT), 2021, pp. 985-989, doi: 10.1109/RTEICT52294.2021.9573867.

[3] R. Liu and Z. Ren, "Application of Yolo on Mask Detection Task," 2021 IEEE 13th International Conference on Computer Research and Development (ICCRD), Beijing, China, 2021, pp. 130-136. doi: 10.1109/ICCRD51685.2021.9386366

[4] S. Abbasi, H. Abdi and A. Ahmadi, "A Face-Mask Detection Approach based on YOLO Applied for a New Collected Dataset," 2021 26th International Computer Conference, Computer Society of Iran (CSICC), Tehran, Iran, 2021, pp. 1-6. doi: 10.1109/CSICC52343.2021.9420599

[5] Sun, Xudong, et al. "Face Detection Using Deep Learning: An Improved Faster RCNN Approach." Neurocomputing (Amsterdam), vol. 299, Elsevier B.V, 2018, pp. 42–50, https://doi.org/10.1016/j.neucom.2018.03.030.

[6] Viola, Paul, and Michael J. Jones. "Robust Real-Time Face Detection." International Journal of Computer Vision, vol. 57, no. 2, Kluwer Academic Publishers, 2004, pp. 137–54, https://doi.org/10.1023/B:VISI.0000013087.49260.fb.

[7] Redmon, Joseph, et al. "You Only Look Once: Unified, Real-Time Object Detection." 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2016, pp. 779–88, https://doi.org/10.1109/CVPR.2016.91

[8] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580-587, doi: 10.1109/CVPR.2014.81.

[9] Singh, Sunil, et al. "Face Mask Detection Using YOLOv3 and Faster R-CNN Models: COVID-19 Environment." Multimedia Tools and Applications, vol. 80, no. 13, Springer US, 2021, pp. 19753–68, https://doi.org/10.1007/s11042-021-10711-8.

[10] Yu, Jimin, and Wei Zhang. "Face Mask Wearing Detection Algorithm Based on Improved YOLO-V4." Sensors (Basel, Switzerland), vol. 21, no. 9, MDPI AG, 2021, p. 3263–, https://doi.org/10.3390/s21093263.

[11] "Leading data labelling and annotation tool to speed up AI projects," Kili Technology, 01-Oct-2021. [Online]. Available: https://kili-technology.com/. [Accessed: 22-Nov-2021].

[12] Google colab. [Online]. Available: https://colab.research.google.com/. [Accessed: 25-Nov-2021].

[13] GitHub. [Online]. Available: https://github.com/ultralytics/yolov5. [Accessed: 25-Nov-2021].